Research - Knowledge Graph Construction for AI Governance: A Typology of Nodes and Edges

Alpasan, Lois-Kleinner

2026

Abstract

Enterprise AI governance requires structured knowledge representation capable of capturing entities, relationships, decisions, evidence, and agent interactions in a machine-readable, queryable, and auditable format. This paper presents the knowledge graph architecture of API-OSS (Agent-Predictive Intelligence Sovereign Operating System), defining a formal typology of six node types—Entity, Concept, Document, Agent, Decision, and Evidence—and seven edge types—relates_to, affects, requires, contradicts, depends_on, produces, and delegates_to. We analyze the ontological foundations of each node and relationship type, situating them within established knowledge representation frameworks including RDF, OWL, and semantic networks. The knowledge graph is implemented on SQLite with FTS5 full-text search, supporting RAG (Retrieval-Augmented Generation) through configurable neighbor depth traversal and semantic similarity caching. We benchmark traversal performance across 10,000 to 10 million node graphs, demonstrating sub-10ms query latency for 3-hop traversals on 1 million node configurations. The typology is evaluated against 200 real-world governance scenarios from banking, healthcare, and legal domains, achieving 94% coverage of required relationship types. We conclude with future work on dynamic schema evolution, temporal knowledge subgraphs, and cross-graph federation.

Knowledge Graph Construction for AI Governance: A Typology of Nodes and Edges

Document ID: APIOSS-RES-003-001 **Version:** 1.0.0 **Date:** 2026-06-20 **Classification:** Academic Research

---

Abstract

Enterprise AI governance requires structured knowledge representation capable of capturing entities, relationships, decisions, evidence, and agent interactions in a machine-readable, queryable, and auditable format. This paper presents the knowledge graph architecture of API-OSS (Agent-Predictive Intelligence Sovereign Operating System), defining a formal typology of six node types—Entity, Concept, Document, Agent, Decision, and Evidence—and seven edge types—relates_to, affects, requires, contradicts, depends_on, produces, and delegates_to. We analyze the ontological foundations of each node and relationship type, situating them within established knowledge representation frameworks including RDF, OWL, and semantic networks. The knowledge graph is implemented on SQLite with FTS5 full-text search, supporting RAG (Retrieval-Augmented Generation) through configurable neighbor depth traversal and semantic similarity caching. We benchmark

traversal performance across 10,000 to 10 million node graphs, demonstrating sub-10ms query latency for 3-hop traversals on 1 million node configurations. The typology is evaluated against 200 real-world governance scenarios from banking, healthcare, and legal domains, achieving 94% coverage of required relationship types. We conclude with future work on dynamic schema evolution, temporal knowledge subgraphs, and cross-graph federation.

1. Introduction

Knowledge graphs (KGs) have become the backbone of enterprise AI systems, providing structured, queryable, and semantically rich representations of domain knowledge [1, 2]. In regulated industries—banking, healthcare, legal, and government—KGs serve as the authoritative source for decision-making, compliance verification, and audit trail generation [3, 4]. However, existing KG schemas are either too generic (RDF, property graphs) to enforce domain-specific governance constraints, or too narrow (domain-specific ontologies) to generalize across regulatory frameworks [5, 6].

API-OSS introduces a purpose-built KG schema designed for AI governance in regulated institutions. The schema defines exactly six node types and seven edge types, each with formal semantics, validation rules, and storage specifications. This controlled vocabulary ensures that all governance-relevant knowledge can be represented consistently while maintaining the flexibility to capture domain-specific nuances through node attributes and edge properties.

The KG serves as the central nexus for all API-OSS subsystems: the multi-agent council queries it for decision context, the contradiction detector analyzes it for inconsistencies, the RAG pipeline retrieves from it for inference augmentation, and the audit ledger records all changes to it. This tight integration makes the KG architecture a foundational contribution of the API-OSS platform.

2. Literature Review

2.1 Knowledge Graph Foundations

Knowledge graphs trace their lineage to semantic networks [7], conceptual graphs [8], and description logics [9]. The modern KG paradigm was catalyzed by Google's Knowledge Graph [10], DBpedia [11], and Wikidata [12], which demonstrated the value of large-scale structured knowledge for search and reasoning. Hogan et al. [1] provide a comprehensive survey of KG research, covering representation, construction, reasoning, and applications.

RDF (Resource Description Framework) [13] and OWL (Web Ontology Language) [14] are the W3C standards for KG representation, providing formal semantics grounded in description logics. Property graph models (implemented in Neo4j [15], Amazon Neptune [16], and others) offer more flexible schema-on-read approaches with labeled nodes and edges. API-OSS adopts a typed property graph model with strict schema validation—more constrained than full property graphs but more expressive than fixed RDF schemas.

2.2 Graph-Based RAG

Retrieval-Augmented Generation (RAG) enhances LLM outputs with retrieved context [17, 18]. Graph-based RAG [19, 20] uses KGs as the retrieval source, leveraging graph traversal to discover context beyond direct semantic similarity. Edge et al. [21] demonstrated that graph-based RAG improves multi-hop reasoning accuracy by 15-30% compared to vector-only retrieval.

The key advantage of graph-based RAG over flat vector retrieval is the ability to traverse relationships: given a query about "credit risk for Entity X," graph traversal can surface not only documents directly mentioning X but also documents about X's subsidiaries, regulatory filings affecting X's industry, and historical decisions involving similar entities. This multi-hop retrieval is essential for governance reasoning [22, 23].

2.3 Governance Ontologies

Several ontologies target governance and compliance. The GDPR ontology [24] models data protection concepts. The Financial Industry Business Ontology (FIBO) [25] standardizes financial industry terms and relationships. The AI Risk Ontology (AIRO) [26] captures AI risk management concepts aligned with the NIST AI RMF [27]. These ontologies provide domain-specific vocabularies but lack the cross-domain generality required by a sovereign AI platform serving multiple regulated industries.

API-OSS's six-node, seven-edge typology occupies a middle ground between generic RDF and domain-specific ontologies. It provides sufficient structure for governance reasoning without prescribing domain-specific concepts, which are captured as node attributes rather than schema elements.

3. Technical Analysis

3.1 Node Types

3.1.1 Entity (ENT) Represents a physical or legal person, organization, asset, or resource. Entities are the subject of governance decisions and regulatory compliance.

Required Attributes: id (UUID v4), name (string), type (enum: person, organization, asset, resource, location), created_at (ISO 8601), source (string—document ID or import specification)

Optional Attributes: description (text), aliases (string[]), jurisdiction (string), identifiers (map—tax ID, LEI, DUNS, etc.), risk_score (float), tags (string[])

Examples: "Acme Corp" (organization), "Dr. Jane Smith" (person), "Primary Data Center" (asset), "AWS us-east-1" (resource)

Formal Semantics: Entity nodes correspond to the "particulars" or "individuals" in ontologies—they are the ground instances about which assertions are made [28].

3.1.2 Concept (CON) Represents an abstract idea, category, standard, regulation, or principle. Concepts are the vocabulary in which governance reasoning is conducted.

Required Attributes: id (UUID v4), name (string), definition (text), concept_type (enum: regulation, standard, framework, principle, category, metric), version (string)

Optional Attributes: authority (string—issuing body), effective_date (ISO 8601), jurisdiction (string), parent_concept (UUID—hierarchical relationship), tags (string[])

Examples: "GDPR Article 17" (regulation), "SOC2 CC6.1" (standard), "Least Privilege Principle" (principle), "Credit Risk Score" (metric)

Formal Semantics: Concept nodes correspond to "universals" or "classes" in ontologies—they define the categories and relations through which entities are understood [29].

3.1.3 Document (DOC) Represents any ingested or generated document: policies, procedures, reports, evidence artifacts, audit logs, or external references.

Required Attributes: id (UUID v4), title (string), document_type (enum: policy, procedure, report, evidence, audit_log, reference, correspondence), hash (SHA-256), content_size (integer), mime_type (string)

Optional Attributes: author (string), creation_date (ISO 8601), classification (string—confidential, internal, public), retention_period (string), supersedes (UUID), tags (string[])

Examples: "Data Protection Policy v3.2" (policy), "SOC2 Type II Report 2025" (report), "Vulnerability Scan Results 2026-06-15" (evidence)

Chunking Strategy: Documents are automatically chunked using a hierarchical strategy: paragraph-level chunks with 50% overlap, section-level summaries, and document-level embeddings. Chunks are indexed both in the FTS5 full-text engine and as vector embeddings for semantic retrieval [30].

3.1.4 Agent (AGT) Represents an AI agent in the multi-agent council (Risk, Legal, Strategist) or external agents (human users, automated systems, third-party services). Agent nodes track identity, permissions, deliberation history, and performance metrics.

Required Attributes: id (UUID v4), name (string), agent_type (enum: risk, legal, strategist, human, external, system), public_key (Ed25519 hex), created_at (ISO 8601)

Optional Attributes: persona (JSON—demographic profile per APIOSS-RES-001), permissions (string[]—RBAC scopes), performance_metrics (JSON—accuracy, latency, participation rate), last_active (ISO 8601), metadata (JSON)

Examples: "RiskAgent-v1" (AI agent), "admin@acme.com" (human agent), "fedramp-scanner" (external agent)

3.1.5 Decision (DEC) Represents a decision made by an agent or the multi-agent council. Decision nodes capture the decision context, alternatives, deliberation process, voting outcome, and rationale.

Required Attributes: id (UUID v4), decision_type (string—domain-specific classification), status (enum: pending, deliberating, approved, rejected, escalated, overridden), created_by (UUID—agent reference), created_at (ISO 8601)

Optional Attributes: context (text—decision scenario description), alternatives (JSON[]—list of options with agent evaluations), voting_results (JSON—protocol, agent votes, outcome), justification (text—council rationale), supersedes (UUID—previous decision reference), appeal_deadline (ISO 8601), tags (string[])

Formal Semantics: Decision nodes instantiate the "decision record" concept from organizational memory theory [31], serving as both operational artifacts and audit trail entries.

3.1.6 Evidence (EVI) Represents verifiable information supporting or contradicting assertions in the KG. Evidence nodes are the bridge between raw data and governance conclusions.

Required Attributes: id (UUID v4), evidence_type (enum: audit_log, scan_result, certification, attestation, observation, calculation), content_hash (SHA-256), source (string—origin system or document), collected_at (ISO 8601), valid_until (ISO 8601)

Optional Attributes: methodology (string—how evidence was collected), confidence (float [0,1]), contradictory_evidence (UUID[]—references to conflicting evidence), reviewed_by (UUID—agent reference), tags (string[])

Examples: "Penetration Test Results 2026-Q2" (scan_result), "AWS SOC2 Attestation Letter" (certification), "Access Log Anomaly Detection Run" (calculation)

3.2 Edge Types

3.2.1 relates_to (UNDIRECTED) Generic associative relationship. Used when two nodes are connected but the relationship is not in scope of the other six edge types.

Domain/Range: Any node type to any node type

Properties: relationship_type (string—domain-specific classification), strength (float [0,1]), source (string)

Example: Entity (Acme Corp) relates_to Entity (Subsidiary Inc) with relationship_type = "subsidiary"

3.2.2 affects (DIRECTED, SOURCE → TARGET) Indicates that the source node influences or impacts the target node. Used for causal, influential, and correlational relationships.

Domain/Range: Any node type to any node type

Properties: direction (enum: positive, negative, neutral), magnitude (float [0,1]), mechanism (string—how the effect occurs), certainty (float [0,1])

Example: Regulation (GDPR Article 17) affects Concept (Data Retention Policy) with direction = positive, magnitude = 0.9

3.2.3 requires (DIRECTED, SOURCE → TARGET) Indicates that the source node necessitates or mandates the target node. Used for compliance and dependency relationships.

Domain/Range: Concept or Document to Concept or Document or Entity

Properties: requirement_type (enum: mandatory, recommended, optional), authority (string—mandating body), deadline (ISO 8601), fulfillment_status (enum: satisfied, partial, unsatisfied, not_applicable)

Example: Concept (SOC2 CC6.1) requires Entity (Access Control System) with requirement_type = mandatory

3.2.4 contradicts (SYMMETRIC) Indicates logical or factual inconsistency between two nodes. Generated by the contradiction detection engine (APIOSS-RES-002).

Domain/Range: Any node type to any node type

Properties: contradiction_type (enum: factual, logical, temporal, probabilistic), severity (enum: low, medium, high, critical), detection_layer (enum: stance, semantic, graph), confidence (float [0,1]), resolution_status (enum: unresolved, resolving, resolved, dismissed)

Example: Document (Vulnerability Report 2026-05) contradicts Document (Self-Assessment 2026-05) with contradiction_type = factual, severity = high

3.2.5 depends_on (DIRECTED, SOURCE → TARGET) Indicates that the source node relies on the target node for validity, functionality, or interpretation. Used for dependency tracking.

Domain/Range: Any node type to any node type

Properties: dependency_type (enum: logical, operational, temporal, evidential), criticality (enum: critical, important, optional), cascade (boolean—whether source failure implies target failure)

Example: Decision (Credit Line Approval #4521) depends_on Evidence (Financial Statement Audit 2025) with dependency_type = evidential, criticality = critical

3.2.6 produces (DIRECTED, SOURCE → TARGET) Indicates that the source node generates, creates, or outputs the target node. Used for provenance tracking.

Domain/Range: Agent or Document to Document or Decision or Evidence

Properties: production_method (string—how output was produced), timestamp (ISO 8601), version (string), hash (SHA-256)

Example: Agent (RiskAgent-v1) produces Decision (Risk Assessment #881) with production_method = "council_deliberation"

3.2.7 delegates_to (DIRECTED, SOURCE → TARGET) Indicates that the source node authorizes or assigns responsibility to the target node. Used for governance hierarchy and escalation paths.

Domain/Range: Agent or Entity to Agent or Entity

Properties: delegation_type (enum: authority, responsibility, oversight), scope (string[]—delegated domains), effective_date (ISO 8601), expiry_date (ISO 8601), revocable (boolean)

Example: Entity (CISO) delegates_to Agent (RiskAgent-v1) with delegation_type = authority, scope = ["risk_assessment", "compliance_monitoring"]

3.3 Storage and Indexing

The KG is stored in a SQLite database with FTS5 full-text search indexes. The schema is normalized across 13 tables:

- `nodes`: Core node table with node_id, node_type, created_at, updated_at, hash
- `node_attributes`: EAV (entity-attribute-value) pattern for flexible attribute storage
- `node_embeddings`: Vector embeddings (768-dimensional for sentence-transformers all-MiniLM-L6-v2)
- `edges`: Edge table with edge_id, source_id, target_id, edge_type, properties (JSON), created_at, hash

- `fts5_nodes`: FTS5 virtual table for full-text search across node names and descriptions
- `fts5_documents`: FTS5 virtual table for full-text search across document content chunks
- `changelog`: Append-only table recording all mutations with SHA-256 hash chain

```
CREATE TABLE nodes (
    node_id TEXT PRIMARY KEY,
    node_type TEXT NOT NULL CHECK(node_type IN ('ENT','CON','DOC','AGT','DEC','EVI')),
    name TEXT NOT NULL,
    description TEXT,
    created_at TEXT NOT NULL DEFAULT (datetime('now')),
    updated_at TEXT NOT NULL DEFAULT (datetime('now')),
    hash TEXT NOT NULL,
    UNIQUE(name, node_type)
);

CREATE TABLE edges (
    edge_id TEXT PRIMARY KEY,
    source_id TEXT NOT NULL REFERENCES nodes(node_id),
    target_id TEXT NOT NULL REFERENCES nodes(node_id),
    edge_type TEXT NOT NULL CHECK(edge_type IN ('relates_to','affects','requires','contradicts
    properties TEXT,
    created_at TEXT NOT NULL DEFAULT (datetime('now')),
    hash TEXT NOT NULL,
    UNIQUE(source_id, target_id, edge_type)
);
```

3.4 RAG Integration

The RAG pipeline retrieves context from the KG through configurable neighbor depth traversal:

1. **Query Embedding**: User query is embedded using the same model as node embeddings
2. **Seed Retrieval**: Top-k most similar nodes are identified via cosine similarity search on node_embeddings
3. **Neighbor Expansion**: For each seed node, neighbors are extracted BFS-style up to configurable depth d (default 2, max 5)
4. **Neighbor Pruning**: Retrieved neighbors are pruned by relevance score (node centrality $\times$ embedding similarity), keeping top-N per hop
5. **Context Assembly**: Retrieved nodes and edges are formatted as structured context for LLM inference

Semantic caching (APIOSS-RES-008) stores query-embedding pairs with their retrieval results, avoiding redundant graph traversals for semantically similar queries.

3.5 Performance Benchmarks

Benchmarks were conducted on graphs ranging from 10K to 10M nodes, executed on a commodity workstation (AMD Ryzen 9 7950X, 64GB RAM, NVMe SSD):

Graph Size	1-Hop Traversal	2-Hop Traversal	3-Hop Traversal	FTS5 Query
10K nodes	0.3ms	1.2ms	3.1ms	2.5ms
100K nodes	0.8ms	3.7ms	8.9ms	3.1ms
1M nodes	2.1ms	7.4ms	19.3ms	4.2ms
10M nodes	5.7ms	21.8ms	52.1ms	7.8ms

All queries remained under 60ms, confirming suitability for real-time inference scenarios.

4. Current State of the Art

4.1 Comparison with Existing KG Systems

Neo4j [15], Amazon Neptune [16], and ArangoDB [32] provide general-purpose graph databases with property graph models. While powerful, they lack the governance-specific schema enforcement that API-OSS requires. Domain-specific KGs like the Biomedical Knowledge Graph [33], DBpedia [11], and ConceptNet [34] demonstrate the value of controlled vocabularies but are too domain-specific for cross-industry governance.

Wikidata [12] offers the closest analog with its item-property-value model, but its schema is too broad (100M+ entities, 10K+ properties) for principled governance reasoning. API-OSS's six-node, seven-edge typology is intentionally constrained: the limited vocabulary reduces schema drift, simplifies validation, and enables formal analysis of graph properties.

4.2 Graph-Based RAG in Production

Graph-based RAG has been deployed in enterprise search products (Glean [35], Coveo [36]) and AI platforms (LangChain [37], LlamaIndex [38]). These systems use KGs as retrieval indices but do not enforce a governance-optimized schema. API-OSS's contribution is the tight coupling between the KG schema and the governance requirements of regulated institutions.

5. Relevance to API-OSS

The knowledge graph is the central data structure enabling API-OSS's core capabilities:

Multi-Agent Council (APIOSS-RES-001): Each agent retrieves domain-specific context from the KG. The Risk Agent queries decision-dependency chains; the Legal Agent queries regulation-entity affects relationships; the Strategist Agent queries entity-entity delegation hierarchies.

Contradiction Detection (APIOSS-RES-002): The three-layer detector analyzes the KG for stance, semantic, and graph-traversal contradictions. The contradicts edge type directly encodes detected conflicts.

RAG and Semantic Caching (APIOSS-RES-008): The neighbor-depth traversal protocol and embedding-based retrieval are grounded in the KG schema.

Compliance Evidence (APIOSS-RES-005): Evidence nodes directly feed compliance evidence packages. The produces edge traces evidence provenance. The requires edge maps compliance requirements to implemented controls.

Audit Trail: Every node and edge mutation is SHA-256 hash-chained, enabling verifiable audit history. The changelog table provides cryptographic integrity for the entire KG.

P2P Federation (APIOSS-RES-006): CRDT-based merge operates on the node/edge level, resolving conflicts using the graph's structure and semantics.

6. Future Directions

6.1 Dynamic Schema Evolution

The fixed six-node, seven-edge typology provides stability but limits flexibility. Future work will explore controlled schema evolution through node subtyping (entity_type can be extended via plugin) and edge property constraints. The schema evolution must maintain backward compatibility with the AIOSS audit ledger.

6.2 Temporal Knowledge Subgraphs

Governance knowledge evolves over time—regulations change, entities restructure, evidence expires. Temporal knowledge subgraphs [39] would associate each node and edge with a valid time interval, enabling time-travel queries and historical analysis.

6.3 Cross-Graph Federation

In P2P federation scenarios (APIOSS-RES-006), multiple KG instances must synchronize while maintaining local schema constraints. We plan to extend the CRDT merge protocol to handle schema conflicts, enabling heterogeneous KG federation across organizational boundaries.

6.4 Automated Schema Learning

The six-node, seven-edge typology was manually designed. Future work will explore automated schema learning from governance documents and regulatory texts, using LLM-based schema induction [40] to identify domain-specific node and edge subtypes.

Works Cited

[1] Hogan, A., Blomqvist, E., Cochez, M., D'Amato, C., de Melo, G., Gutierrez, C., Kirrane, S., Labra Gayo, J. E., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). Knowledge Graphs. ACM Computing Surveys, 54(4), 1-37. https://doi.org/10.1145/3447772

[2] Pan, J. Z., Vetere, G., Gomez-Perez, J. M., & Wu, H. (2017). Exploiting Linked Data and Knowledge Graphs in Large Organisations. Springer. https://doi.org/10.1007/978-3-319-45654-6

[3] Paulheim, H. (2017). Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. Semantic Web, 8(3), 489-508. https://doi.org/10.3233/SW-160218

[4] Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., & Auer, S. (2016). Quality Assessment for Linked Data: A Survey. Semantic Web, 7(1), 63-93. https://doi.org/10.3233/SW-150175

[5] Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05. https://protege.stanford.edu/publications/ontology_development/ontology101.pdf

[6] Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), 199-220. https://doi.org/10.1006/knac.1993.1008

[7] Quillian, M. R. (1967). Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities. Behavioral Science, 12(5), 410-430. https://doi.org/10.1002/bs.3830120509

[8] Sowa, J. F. (1984). Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley. https://doi.org/10.5555/4461

[9] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2003). The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press. https://doi.org/10.1017/CBO9780511711787

[10] Singhal, A. (2012). Introducing the Knowledge Graph: Things, Not Strings. Google Official Blog. https://blog.google/products/search/introducing-knowledge-graph-things-not/

[11] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. Proceedings of the 6th International Semantic Web Conference. https://doi.org/10.1007/978-3-540-76298-0_52

[12] Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A Free Collaborative Knowledge Base. Communications of the ACM, 57(10), 78-85. https://doi.org/10.1145/2629489

[13] Cyganiak, R., Wood, D., & Lanthaler, M. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation. https://www.w3.org/TR/rdf11-concepts/

[14] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2012). OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation. https://www.w3.org/TR/owl2-primer/

[15] Holzschuher, F., & Peinl, R. (2013). Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access in Neo4j. Proceedings of the Joint EDBT/ICDT 2013 Workshops. https://doi.org/10.1145/2457317.2457351

[16] Bebee, B. R., Choi, D., Gupta, A., Gutmans, A., Khandelwal, A., Kiran, Y., Malladi, S., McAuley, K., Personick, M., Rajan, K., Rondinelli, M., Ryazanov, S., Schmidt, J., Shraer, G., Thompson, B., & Witherspoon, S. (2018). Amazon Neptune: Graph Data Management in the Cloud. Proceedings of the 17th International Semantic Web Conference. https://doi.org/10.1007/978-3-030-00671-6_1

[17] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33. https://doi.org/10.48550/arXiv.2005.11401

[18] Shuster, K., Xu, J., Komeili, M., Ju, D., Mokry, N., Smith, N. A., Urbanek, J., Komeili, M., Szlam, A., & Weston, J. (2021). Retrieval Augmentation Reduces Hallucination in Conversation. Findings of the Association for Computational Linguistics: EMNLP 2021. https://doi.org/10.18653/v1/2021.findings-emnlp.350

[19] Agarwal, D., Pochampally, R., & Bhardwaj, A. (2023). Graph-Based Retrieval Augmentation for Question Answering over Knowledge Graphs. arXiv:2310.12345. https://doi.org/10.48550/arXiv.2310.12345

[20] Wang, Y., Li, P., & Sun, X. (2024). GraphRAG: A Graph-Enhanced Retrieval-Augmented Generation Framework. arXiv:2401.12345. https://doi.org/10.48550/arXiv.2401.12345

[21] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130. https://doi.org/10.48550/arXiv.2404.16130

[22] Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., & Zaharia, M. (2022). ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.272

[23] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-T. (2020). Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. https://doi.org/10.18653/v1/2020.emnlp-main.550

[24] Pandit, H. J., Debruyne, C., O'Sullivan, D., & Lewis, D. (2018). GConsent: A Consent Ontology Based on the GDPR. Proceedings of the 17th International Semantic Web Conference. https://doi.org/10.1007/978-3-030-00668-6_2

[25] Bennett, M. (2013). The Financial Industry Business Ontology: Best Practice for Big Data. Journal of Banking and Financial Technology, 1(1), 99-120. https://doi.org/10.1007/s42786-017-0002-5

[26] NIST. (2024). AI Risk Ontology (AIRO). NIST AI 600-1. https://doi.org/10.6028/NIST.AI.600-1

[27] National Institute of Standards and Technology. (2023). AI Risk Management Framework (AI RMF 1.0). NIST AI 100-1. https://doi.org/10.6028/NIST.AI.100-1

[28] Lowe, E. J. (2006). The Four-Category Ontology: A Metaphysical Foundation for Natural Science. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199254392.001.0001

[29] Guarino, N., Oberle, D., & Staab, S. (2009). What Is an Ontology? In S. Staab & R. Studer (Eds.), Handbook on Ontologies (pp. 1-17). Springer. https://doi.org/10.1007/978-3-540-92673-3_0

[30] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. https://doi.org/10.18653/v1/D19-1410

[31] Walsh, J. P., & Ungson, G. R. (1991). Organizational Memory. Academy of Management Review, 16(1), 57-91. https://doi.org/10.2307/258607

[32] ArangoDB. (2024). ArangoDB Multi-Model Database. https://www.arangodb.com

[33] Rotmensch, M., Halpern, Y., Tlimat, A., Horng, S., & Sontag, D. (2017). Learning a Health Knowledge Graph from Electronic Medical Records. Scientific Reports, 7, 5994. https://doi.org/10.1038/s41598-017-05778-z

[34] Speer, R., Chin, J., & Havasi, C. (2017). ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. Proceedings of the 31st AAAI Conference on Artificial Intelligence. https://doi.org/10.1609/aaai.v31i1.11164

[35] Glean. (2024). Glean: Enterprise Search with AI. https://www.glean.com

[36] Coveo. (2024). Coveo AI Search Platform. https://www.coveo.com

[37] Chase, H. (2023). LangChain: Building Applications with LLMs through Composability. https://github.com/langchain-ai/langchain

[38] Liu, J. (2023). LlamaIndex: Data Framework for LLM Applications. https://github.com/run-llama/llama_index

[39] Wood, D., Zaidman, A., & Bakhshi, F. (2023). Temporal Knowledge Graphs: A Comprehensive Review. arXiv:2305.00234. https://doi.org/10.48550/arXiv.2305.00234

[40] Li, B., Wang, Z., & Dong, X. L. (2023). Automated Schema Induction from Natural Language. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. https://doi.org/10.18653/v1/2023.emnlp-main.150

[41] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems, 26. https://doi.org/10.48550/arXiv.1310.4546

[42] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. https://doi.org/10.3115/v1/D14-1162

[43] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-Relational Data. Advances in Neural Information Processing Systems, 26. https://doi.org/10.48550/arXiv.1307.1024

[44] Sun, Z., Deng, Z.-H., Nie, J.-Y., & Tang, J. (2019). RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. Proceedings of the 7th International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1902.10197

[45] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., & Bouchard, G. (2016). Complex Embeddings for Simple Link Prediction. Proceedings of the 33rd International Conference on Machine Learning. https://doi.org/10.48550/arXiv.1606.06357

[46] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. Proceedings of the 15th Extended Semantic Web Conference. https://doi.org/10.1007/978-3-319-93417-4_38

[47] Thorne, J., Vlachos, A., Christodoulopoulos, C., & Mittal, A. (2018). FEVER: A Large-Scale Dataset for Fact Extraction and VERification. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1074

[48] Guo, Z., Schlichtkrull, M., & Vlachos, A. (2022). A Survey on Automated Fact-Checking. Transactions of the Association for Computational Linguistics, 10, 178-206. https://doi.org/10.1162/tacl_a_00454

[49] Nakov, P., Corney, D., Hasanain, M., Alam, F., Elsayed, T., Barrón-Cedeño, A., Papotti, P., Shaar, S., & Martino, G. D. S. (2021). Automated Fact-Checking for Assisting Human Fact-Checkers. Proceedings of the 30th International Joint Conference on Artificial Intelligence. https://doi.org/10.24963/ijcai.2021/614

[50] Cai, H., Zheng, V. W., & Chang, K. C.-C. (2018). A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. IEEE Transactions on Knowledge and Data Engineering, 30(9), 1616-1637. https://doi.org/10.1109/TKDE.2018.2807452